Topic-based Selectivity Estimation for Hybrid Queries over RDF Graphs

Andreas Wagner
AIFB, KIT
Karlsruhe, Germany
a.wagner@kit.edu

Veli Bicer
IBM Research, Smarter Cities Technology Centre
Damastown Industrial Estate, Dublin, Ireland
velibice@ie.ibm.com

Thanh Duc Tran
AIFB, KIT
Karlsruhe, Germany
ducthanh.tran@kit.edu

ABSTRACT

The Resource Description Framework (RDF) has become an accepted standard for describing entities on the Web. At the same time, many RDF descriptions today are *text-rich* – besides *structured* data, they also feature large portions of *unstructured* text. Such semi-structured data is frequently queried using predicates matching structured data, combined with string predicates for textual constraints: *hybrid queries*. Evaluating hybrid queries efficiently requires effective means for *selectivity estimation*. Previous works on selectivity estimation, however, target either structured or unstructured data alone. In contrast, we study the problem in a *uniform manner* by exploiting a topic model as data synopsis, which enables us to accurately capture *correlations between structured and unstructured data*. Relying on this synopsis, our novel topic-based approach (`TopGuess`) uses as small, fine-grained *query-specific* Bayesian network (BN). In experiments on real-world data we show that the query-specific BN allows for great improvements in estimation accuracy. Compared to a baseline relying on PRMs we could achieve a gain of 20%. In terms of efficiency `TopGuess` performed comparable to our baselines.

1. INTRODUCTION

The amount of RDF available on the Web today, such as Linked Data, RDFa and Microformats, is large and rapidly increasing. RDF data contains descriptions of entities, with each description being a set of *triples*: $\{\langle s, p, o\rangle\}$. A triple associates an entity (subject) s with an *object* o via a *predicate* p. A set of triples forms a data graph, cf. Fig. 1.

Text-rich Data. Many RDF entity descriptions are *text-rich*, i.e., contain large amounts of unstructured data. On the one hand, structured RDF often comprises textual data via predicates such as `comment` or `description`. Well-known examples include datasets such as DBpedia[1] or IMDB[2]. On the other hand, unstructured (Web) documents are frequently annotated with structured data, e.g., RDFa or Microformats.[3] Such interlinked documents can be seen as an RDF graph with document texts as objects.

Conjunctive Hybrid Queries. The standard language for querying RDF is SPARQL, which at its core features *conjunctive queries*. Such queries comprise a conjunction of *query predicates* $\langle s, p, o\rangle$. Here, s, p and o may refer to a *variable* or a *constant*, i.e., an entity, a predicate or an object in the data. Given text-rich data, a query predicate can either match structured data or *keywords* in unstructured texts (*hybrid query*). The latter is commonly handled via string predicates modeling textual constraints. Processing conjunctive queries involves an optimizer, which relies on *selectivity estimates* to construct an "optimal" query plan, thereby minimizing intermediate results.

Selectivity Estimation. Aiming at a low space and time complexity, selectivity estimation is based on *data synopses*, which approximately capture data value distributions. For simplicity assumptions are commonly employed:

(1) The *uniform distribution assumption* implies that all objects for a predicate are equally probable. For instance, the probability for an entity x having `name` "Mel Ferrer" is $P(X_{name} = \text{"Mel}\ldots\text{"}) \approx 1/|\Omega(X_{name})| = 1/4$ (Fig. 1). Thus, the probability for a query predicate $\langle x, name, \text{"Mel}\ldots\text{"}\rangle$ is $1/4$. Given entities with a common `name` this leads to misestimates.

(2) Consider a second predicate $\langle x, comment, \text{"Audrey Hepburn was}\ldots\text{"}\rangle$ (probability 1), the *predicate value independence assumption* states that the two object values are independent. Thus, $P(X_{name} = \text{"Mel}\ldots\text{"}, X_{comment} = \text{"Audrey}\ldots\text{"}) \approx P(X_{name} = \text{"Mel}\ldots\text{"}) \cdot P(X_{comment} = \text{"Audrey}\ldots\text{"}) = 1/4 \cdot 1$. However, there is no entity having both predicates `name` and `comment`. Such misestimates are due to correlations: Given a value for `name`, a particular value for `comment` is more or less probable.

(3) The *join predicate independence* assumption (special case of (2)) implies that the existence of a predicate is independent of the value respectively existence of another predicate. For instance, the existence of `comment` and any value for `name` would be assumed independent. Such a simplification results in errors, as `comment` only occurs with `name` "Audrey Kathleen Hepburn".

Such assumptions lead to severe misestimates, which in turn results in costly query plans being constructed. Thus, there

[1] `http://dbpedia.org`
[2] `http://www.linkedmdb.org`
[3] `http://www.webdatacommons.org`

is a strong need for *efficient and effective* data synopses (discussed in Sect. 3).

Synopses for Un-/Structured Data. A large body of work has been devoted to synopses for structured data. Assumption (1) is addressed by frequency statistics embedded in, e.g., histograms [15]. Assumption (2)/(3) require an approximation of joint distributions of multiple random variables via join synopses [1], tuple-graph synopses [17] or probabilistic relational models [10, 19, 20] (PRM).

In context of text-rich data, a key challenge is that random variables have *large sample spaces*. That is, string predicates comprise keywords, which match any text value that *contains* such keywords. Thus, sample spaces (e.g., $\Omega(X_{name})$ Fig. 1) must comprise all words and phrases (sequences of words) contained in text values. Addressing this issue, *string synopses* summarizing string sets, such as $\Omega(X_{name})$, have been proposed. Synopses based on pruned suffix trees, Markov tables, clusters or n-grams, have received much attention [5, 11, 21]. However, such approaches solely address one single string predicate, e.g., $\langle x, name, \text{“Mel...”}\rangle$. In fact, they do not consider any kind of correlations among multiple string predicates and/or structured query predicates.

On the other hand, synopses for structured data do not summarize texts. In fact, [10] noted the need for summaries over such “large” sample spaces, and [20] addressed this issue by a “loose coupling” of string synopses and PRMs. However, no *uniform framework* for capturing correlations between unstructured and structured data elements has been proposed. In particular, [20] lacks a uniform synopsis construction, and suffers from an “information loss” due to inaccurate string summaries.

Relational Topic Models. Recently, multiple works targeting *relational topic models* [4, 24, 23] have extended traditional topic models to not only reflect text, but also structure information. In our selectivity estimation approach (`TopGuess`), we exploit this line of work, in order to have *one single synopsis for text and structure information*. That is, we use Topical Relational Models (TRM), which have been proposed as means for bridging the gap between text-oriented topic models (e.g., LDA [3]) and structure-focused relational learning techniques (e.g., PRMs) [23].

In the following, we will show that using a relational topic model synopses, we have a number of key advantages over traditional selectivity estimation solutions. In particular, `TopGuess` does not require an expensive “unrolling” of random variables as in PRM-approaches, in order to compute query predicate probabilities. In fact, `TopGuess` can manage its synopsis from disk and does not use in-memory inferencing. In terms of space, `TopGuess` achieves a linear space complexity in the size of its string synopsis.

Contributions. Let us outline our contributions in more details: (1) In this work, we present a general framework for selectivity estimation for hybrid queries over text-rich RDF and outline its main requirements. In particular, we discuss conceptual drawbacks of PRM-based solutions [10, 19, 20] in light of these requirements. (2) We introduce a novel approach (`TopGuess`), which utilizes a *relational topic model* based synopsis, to summarize text-rich graph-structured data. In particular, we show how `TopGuess` estimates the selectivity, without the need for inferencing, via a *query-specific*, fine-grained, yet still compact BN. (4) In our experiments, we use baselines relying on independence between query predicates (assumptions (2) and (3)) and PRM-based approaches. Our results suggest that accuracy of selectivity estimates can be greatly improved, if dependencies between un-/structured data is taken into account.

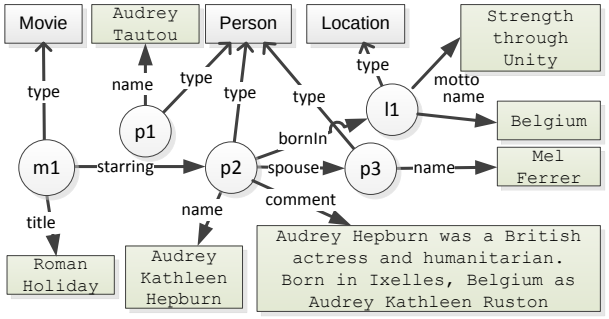


Figure 1: RDF data graph about Audrey Hepburn and her movie “Roman Holiday”.

Outline. First, in Sect. 2 we outline preliminaries. Sect. 3 introduces a general framework for selectivity estimation. We introduce our novel `TopGuess` approach in Sect. 4. In Sect. 5 we present evaluation results, before we outline related work in Sect. 6, and conclude with Sect. 7.

2. PRELIMINARIES

We use RDF as data and conjunctive queries as query model:

Data. RDF data is following a graph model as follows: Let ℓ_a (ℓ_r) denote a set of attribute (relation) labels. Data is given by a directed labeled graph $\mathcal{G} = (V, E, \ell_a, \ell_r)$, where V is the disjoint union $V = V_E \uplus V_A \uplus V_C$ of *entity* nodes V_E, *attribute value* nodes V_A and *class* nodes V_C. Edges (triples) $E = E_R \uplus E_A \uplus type$ are a disjoint union of *relation* edges E_R and *attribute* edges E_A. Let relation edges connect entity nodes, i.e., $\langle s, r, o\rangle \in E_R$ iff $s, o \in V_E$ and $r \in \ell_r$), and attribute edges connect an entity with an attribute value, $\langle s, a, o\rangle \in E_A$ iff $s \in V_E, o \in V_A$ and $a \in \ell_a$. The “special” edge $\langle s, type, o\rangle \in E$, $s \in V_E$ and $o \in V_C$, models entity s belonging to class o. For sake of simplicity we omit further RDF features, such as blank nodes or class/predicate hierarchies. If an attribute value $o \in V_A$ contains text, we conceive it as a *bag-of-words*. Further, we say that a vocabulary W comprises all such bags-of-words $\in V_A$. An Example is given in Fig. 1.

Conjunctive Hybrid Queries. Conjunctive queries represent the *basic graph pattern* (BGP) feature of SPARQL. Generally speaking, conjunctive queries are a core fragment of many structured query languages (e.g., SQL or SPARQL). We use a particular type of conjunctive queries, *hybrid queries*, being a conjunction of query predicates of the form: A query Q, over a data graph G, is a directed labeled graph $G_Q = (V_Q, E_Q)$ where V_Q is the disjoint union $V = V_{Q_V} \uplus V_{Q_C} \uplus V_{Q_K}$ of *variable* nodes (V_{Q_V}), *constant* nodes (V_{Q_C}) and *keyword* nodes (V_{Q_K}), where $o \in V_{Q_K}$ is a user-defined keyword. For simplicity, in this work we define a keyword node as “one word” occurring in an attribute value. That is, a keyword is one element from a bag-of-words representation of an attribute node. Corresponding to edge types, ℓ_a, ℓ_r, and *type*, we distinguish three kinds of query predicates: *class predicates* $\langle s, type, o\rangle, s \in V_{Q_V}, o \in V_{Q_C}$, *relation predicates* $\langle s, r, o\rangle, s \in V_{Q_V}, o \in V_{Q_C}, r \in \ell_r$ and *string predicates* $\langle s, a, o\rangle, s \in V_{Q_V}, o \in V_{Q_K}, a \in \ell_a$. An example is given in Fig. 2. The query semantics follow those for BGPs: results are subgraphs of a data graph matching the graph pattern captured by the query. The only difference is due to keyword nodes: a value node $o' \in V_A$ matches a keyword node $o'' \in V_{Q_K}$, if words in o' contain all words from o''. Lastly, a BGP query may contain attribute predicates with other value domains such as date and time. In this work, however, we focus on textual attributes.

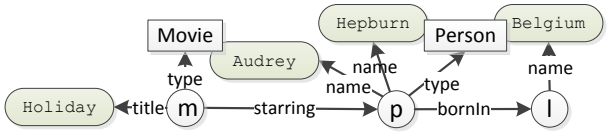


Figure 2: Hybrid Query asking for movies with `title` "Holiday" and `starring` "Audrey Hepburn".

In the following, we introduce two models, which we use as selectivity estimation synopses:

Bayesian Networks. A Bayesian network (BN) is a directed graphical model allowing for a compact representation of joint distributions via its structure and parameters [12]. The structure is a directed acyclic graph, where nodes stand for random variables and edges represent dependencies. Given parents $\mathbf{Pa}(X_i) = \{X_j, \ldots, X_k\}$, a random variable X_i is dependent on $\mathbf{Pa}(X_i)$, but *conditionally independent* of all non-descendant random variables.

Example 1. See Fig. 3-a for an BN. Here, e.g., X_{title} and X_{movie} denote random variables. The edge $X_{movie} \rightarrow X_{title}$ refers to a dependency between the parent variable, $X_{movie} = \mathbf{Pa}(X_{title})$, and the child X_{title}. X_{title} is, however, conditionally independent of all non-descendant variables, e.g., X_{name} or $X_{comment}$.

BN parameters are given by conditional probability distributions (CPDs). That is, each random variable X_i is associated with a CPD capturing the conditional probability $P(X_i|\mathbf{Pa}(X_i))$. An extract of a CPD is shown in Fig. 3-b. The joint distribution $P(X_1, \ldots, X_n)$ can be estimated via the chain rule: $P(X_1, \ldots, X_n) \approx \prod_i P(X_i|\mathbf{Pa}(X_i))$ [12].

Topic Models. Topic models are based on the idea that documents are mixtures of "hidden" topics, where each topic is a probability distribution over words. These topics, constitute abstract clusters of words categorized according to their co-occurrence in documents. More formally, a document collection can be represented by K topics $\mathcal{T} = \{t_i, \ldots, t_K\}$, where each $t \in \mathcal{T}$ is a multinomial distribution of words $P(w \mid t) = \beta_{tw}$ and $\sum_{w \in W} \beta_{tw} = 1$. Here, W represents the vocabulary of individual words appearing in the corpus. This way the entire corpus can be represented as K topics, which leads to a low-dimensional representation of the contained text.

Example 2. Three topics are shown in Fig. 5-c. Every topic assigns a probability (represented by vector $\boldsymbol{\beta}_t$) to each word in the vocabulary, indicating the importance of the word within that topic. For example, word "Belgium" is more important in the third topic (probability $\beta_{tw} = 0.014$), than for the other two topics.

Prominent topic models, e.g., Latent Dirichlet allocation (LDA) [3], are modeled as graphical models (e.g., Bayesian networks). They assume a simple probabilistic procedure, called *generative model*, by which documents can be generated. To generate a document, one chooses a distribution over topics. Then, for each word in that document, one selects a topic at random according to this distribution, and draws a word from that topic. Since initially the topic distribution of the documents and the word probabilities within the topics are unknown (hidden), this process is inverted and the standard Bayesian learning techniques are used to learn the hidden variables and topic parameters (e.g., β_{tw}).

Recently, relational topic models [4, 24, 23] have been introduced for capturing dependencies between words $\in W$ and structural information, such as entity relations and classes. In our work we rely on Topical Relational Models (TRMs) (Fig. 4 + 5), as they are closest to our data model. Further details on TRMs are presented in Sect. 4.

Problem Definition. Given a hybrid query over an RDF graph, we tackle the problem of *effective* and *efficient* selectivity estimation. In particular due to the *contains* semantics of string predicates, we face several issues and challenges, which we also discuss in a formal manner via a requirements analysis in Sect. 3. In the following, let us outline them from an informal point of view.

(1) *Efficiency Issues.* If a vocabulary W of words occurring in attributes $\in V_A$ is large, the data synopsis, e.g., a PRM [10, 19], can grow exponentially and quickly become complex. Recall, a synopsis needs to capture statistics for all correlations among words $\in W$ as well as other structured data (classes and relations). Consider, e.g., predicate $\langle p, name, \text{"Audrey"} \rangle$, a data synopsis may summarize the entity count of bindings for variable p as 2 ($p_{1/2}$ in Fig. 1). However, given a second query predicate, e.g., $\langle p, name, \text{"Hepburn"} \rangle$ or $\langle m, starring, p \rangle$, correlations occur and a synopsis would need store the count of the conjoined query (*join predicate independence* assumption fails). Thus, it has to capture separate counts for `name` "Audrey", i.e., comprise all possible combinations with other words (e.g., "Hepburn") and structural elements (e.g., `starring`). Clearly, creating such a data synopsis is infeasible for text-rich datasets. Intuitively, a synopsis must "scale" well w.r.t. vocabulary W respectively a string synopsis on W (**Req.2**). Further, an selectivity estimation that builds up on the data synopsis must efficiently approximate the probability for a given query (**Req.4+5**).

(2) *Effectiveness Issues.* Space and time efficiency should not be "traded off" for effectiveness. That is, a vocabulary W must be accurately represented within the data synopsis. Thus, a synopsis should strive to capture "all important" correlations between/among words and/or structured data (**Req.1**). Further, if a string synopsis is employed for summarizing vocabulary W, there must not be an *information loss* (**Req.3**). Such a loss occurs, e.g., due to eliminating "less important" strings (e.g., n-gram synopsis [21]) or by capturing multiple strings with one single string synopsis element (e.g., histogram synopsis [11]).

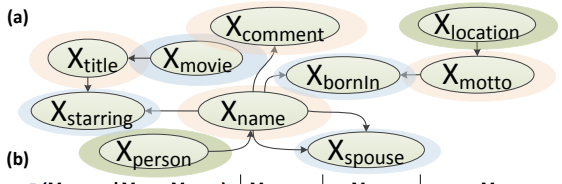
3. ESTIMATION FRAMEWORK

We now introduce a generic *framework for selectivity estimation* over text-rich RDF data and discuss requirements necessary for efficient and effective instantiations.

Framework. Selectivity estimation strategies commonly summarize a data graph $\mathcal{G}$ via a concise *data synopsis*, $\mathcal{S}$, and estimate the selectivity $sel_{\mathcal{G}}(Q)$ by using *an estimation function* $F_{\mathcal{S}}(Q)$ over this synopsis. In the presence of text-rich data, a *string synopsis* function ν is defined as a mapping from the union of words comprised in attribute text values $\in V_A$ (W) and query keyword nodes $\in V_{Q_K}$ to a common representation space, denoted by $\mathcal{C}$. This common space $\mathcal{C}$ has the purpose to *compactly describe* the large set $W \bigcup V_{Q_K}$, while still capturing "as much information" as possible. We define the estimation framework as:

Definition 1 (Estimation Framework)**.** Given a data graph $\mathcal{G}$ and a query Q, an instance of the selectivity estimation framework $sel_{\mathcal{G}}(Q)$ is a tuple $(\mathcal{S}, F_{\mathcal{S}}(Q), \nu)$, where the data synopsis $\mathcal{S}$ represents a summary of the data graph $\mathcal{G}$. The estimation function $F_{\mathcal{S}}(Q)$ approximates the selectivity of query Q using $\mathcal{S}$: $sel_{\mathcal{G}}(Q) \approx F_{\mathcal{S}}(Q)$. ν represents a string synopsis function defined as $\nu : W \bigcup V_{Q_K} \mapsto 2^{\mathcal{C}}$.

Instantiations. The three framework components have been instantiated differently by various approaches. For instance, PRMs [10, 19], graph synopsis [17], or join samples



(b)

$P(X_{starring} \mid X_{title}, X_{name})$	$X_{starring}$	X_{title}	X_{name}
1/9	T	„Roman“	„Audrey“
0	F	„Roman“	„Audrey“
1/9	T	„Roman“	„Hepburn“
0	F	„Roman“	„Hepburn“
1/9	T	„Roman“	„Audrey Kathleen“
0	F	„Roman“	„Audrey Kathleen“
0	T	„Roman“	„Tautou“
1/9	F	„Roman“	„Tautou“
⋮	⋮	⋮	⋮

Figure 3: (a) BN for example data graph (Fig. 1). (b) Extract of CPD for $X_{starring}$.

[1] have been proposed as a data synopsis $\mathcal{S}$. Depending on the synopsis type, different estimation functions were adopted. For example, a function based on Bayesian inferencing [10, 19] or graph matching [17]. Among the many instantiations for $\mathcal{S}$, PRMs [10, 19, 20] are closest to our work. For string synopsis function ν, approaches such as suffix trees, Markov tables, clusters or n-grams can be used [5, 11, 21]. Thus, a space $\mathcal{C}$ could, e.g., comprise of clusters of strings or of histogram buckets. However, most appropriate for the *contains* semantic of string predicates is recent work on n-gram synopses [21]. In particular, this approach does not limit the size of texts in the data, which is a key feature for RDF. Combining a PRM with n-gram string synopses results in a representation space $\mathcal{C}$ comprising n-grams (1-grams to be precise), and resembles previous work in [20]. Here, a random variable X_a, capturing an attribute a in the data graph, has all n-grams occurring in text values associated with a as sample space. Function $F_{\mathcal{S}}(Q)$ can be realized by transforming query predicates $\in Q$ to random variable assignments and calculating their joint probability using BN inferencing [10, 19, 20].

Example 3. Consider the BN in Fig. 3-a with three kinds of random variables: *class* (green), *relation* (blue) and *attribute* (orange) variables. Class random variables, X_c, capture whether or not an entity has a particular class $\in V_C$. For instance, we have a random variable X_{movie} for class `Movie`. Further, a relation variable, X_r, models the event of two entities sharing a relation r. Consider, e.g., $X_{starring}$ referring to the `starring` relation. Each relation random variable has two parents, which correspond to the "source" respectively "target" of that relation. Class and relation random variables are binary: $\Omega(X_r) = \Omega(X_c) = \{\mathbf{T}, \mathbf{F}\}$. Last, attribute assignments are captured via random variables X_a with words as sample space. X_{title}, e.g., has words comprised in movie titles as events.

Requirements for Instantiations. A number of requirements are necessary for an effective and efficient framework instantiation. A synopsis $\mathcal{S}$ should summarize the data in graph $\mathcal{G}$ in the "best way possible". In particular, $\mathcal{S}$ should capture correlations among words, but also correlations between words and classes/relations. In simple terms, independence assumptions (1)-(3), as presented earlier, have to be omitted due to their error prone nature (**Req.1**). The synopsis size is crucial. In fact, a synopsis should aim at a linear space complexity w.r.t. the size of the string synopsis, i.e., $|\mathcal{C}|$ (**Req.2**). Linear space complexity is required to eliminate the exponential growth of a data synopsis w.r.t. its string synopsis (representation space). Accordingly, the need to drastically "reduce" the vocabulary W of words $\in V_A$ via a string synopsis ν should no longer be necessary – even for large datasets. This is crucial as reducing $\mathcal{C}$ always introduces an information loss. An information loss, in turn, directly affects **Req.1**, which aimed at capturing all relevant dependency information (**Req.3**). Since the estimation function $F_{\mathcal{S}}$ operates at runtime, it should be highly efficient (**Req.4**). Last, the time complexity of $F_{\mathcal{S}}$ needs to be independent from the size of the synopsis $\mathcal{S}$ (**Req.5**).

Discussion. As outlined, closest to our work is a framework instantiation utilizing a PRM as data synopsis $\mathcal{S}$ and a n-gram synopsis as string synopsis function ν [20]. Given such an instantiation, correlations are captured via probabilities in CPDs respectively a BN network structure. Learning techniques have been proposed to capture the most crucial correlations efficiently [12]. However, when several values are assigned to the same query predicate respectively random variable representing it, e.g., `name`="Audrey" and `name`="Hepburn" (Fig. 2), an aggregation function must be applied [18], resulting in one single random variable assignment. Thus, dependency information can not be reflected completely. This leads to misestimations, as we observed in our experiments (partially fulfilled **Req.1**). Data synopsis size of a BN is exponential in the string synopsis size, $|\mathcal{C}|$. Thus, **Req.2** is not fulfilled. Further, there is the need to reduce the representation space of ν comprising n-grams, as the size of synopsis $\mathcal{S}$ is strongly affected by the size of $\mathcal{C}$. This, in turn, leads to an information loss (not meeting **Req.3**): in the case of an n-gram string synopsis, reducing $\mathcal{C}$ means to select a subset of n-grams occurring in the text. Note, discarded n-grams may only be estimated via string synopsis heuristics. The probabilities computed from these heuristics may not correspond to the actual probability of the keyword in the query predicate. Our experiments, as well as our previous work [20], show that these information losses lead to significant misestimations. PRM-based data synopses use BN inferencing in order to implement the estimation function $F_{\mathcal{S}}$ [10, 19, 20]. However, it is known that inferencing is $\mathcal{NP}$-hard [7]. Thus, "exact" computation of $F_{\mathcal{S}}$ is not feasible – instead *approximation strategies*, e.g., *Markov Chain Monte Carlo* methods, are used to guarantee an polynomial time complexity of $F_{\mathcal{S}}$ [12] (partially fulfills **Req.4**). Last, BN (PRM) solutions requires expensive computation at runtime. (1) PRM approaches [10, 19, 20] require an *unrolling procedure*: a "query BN" is generated from a template using marginalization. (2) A (query) BN is used for inferencing to compute the query probability. For both such steps, however, computation time is driven by CPD sizes, and thus, synopsis size and complexity of $F_{\mathcal{S}}$ is directly coupled (**Req.5** fails).

4. TOPGUESS

`TopGuess` is another instantiation of our framework (Def. 1): we present a synopsis $\mathcal{S}$ in Sect. 4.1, and an estimation function $F_{\mathcal{S}}$ in Sect. 4.2. Targeting the above requirements, `TopGuess` adheres to several *design decisions*.

First, `TopGuess` utilizes a data synopsis $\mathcal{S}$, which has *linear space complexity* w.r.t. its string synopsis (**Req.2**). In particular, this data synopsis employs *topic models* to obtain a effective representation of correlations between topics and words respectively structural elements, i.e., classes and relations (**Req.1**). In addition, the use of topics allows `TopGuess` to exploit a *complete vocabulary* W as a string synopsis, i.e., W *equals* the common representation space $\mathcal{C}$ of string synopsis ν (**Req.3**). Our synopsis is used to

construct a small BN *only for the current query* at runtime ("query-specific" BN). Essentially, our query-specific BN follows the same intuition as an unrolled BN in a PRM approach [12]. However, as we later show, it is much more *compact* and its construction does *not require expensive marginalization*. Further, we reduce computing the selectivity estimation function F_S, to the task of approximating the joint probability of that query-specific BN – without any inferencing (**Req.4**). In order to better capture correlations among query predicates, `TopGuess` utilizes an assumption as follows: every query variable is modeled as a (hidden) mixture of topics. The query probability can be computed based on the topic mixtures associated with its query variables. In addition, each query predicate separately contributes to the topics of its corresponding query variable. Thus, in contrast to a PRM-based synopsis, there is no need to use an aggregation function, as multiple value assignments to a single random variable are not possible (**Req.1**). `TopGuess` only performs topic mixture learning in its estimation function to enable a fine-grained "conditioning" of query predicates on the topics. However, this learning process is bounded by the query variables. In particular, the learning and, thus, the computation of the estimation function F_S is not affected by the overall data (string) synopsis size (**Req.5**).

4.1 Topic-based Data Synopsis

We exploit a relational topic model [4, 24, 23] as data synopsis $\mathcal{S}$, thereby providing us a *single, uniform synopsis* for structured and unstructured data. In this work, we use *Topical Relational Model* (TRM) [23], as it supports our data model best. Further, TRM parameters may easily be used for calculating query predicate probabilities – as discussed in the following. However, notice that our `TopGuess` approach could be extended to other relational topic models.

Topical Relational Model. Intuitively speaking, a TRM summarizes textual information via a small number of *topics*. Additionally, it *finds correlations* between these topics and structured data elements, i.e., classes and relations. More precisely, a TRM assumes that if entities exhibit structural resemblances (i.e., have similar classes or relations), their words and topics respectively, shall also be similar. Vice versa, given specific words and topics respectively, some structure elements are more probable to be observed than others. For instance, one may observe that words such as "Hepburn" highly correlate with classes like `Person` and `Movie`, as well as with other words like "play" and "role" in the context of a relation `starring`.

A TRM captures correlations between text and structured data via a set of K topics $\mathcal{T} = \{t_i, \ldots, t_K\}$. Each topic $t \in \mathcal{T}$ is a multinomial distribution of words $p(w|t) = \beta_{tw}$ and $\sum_{w \in W} \beta_{tw} = 1$. As before, W represents the vocabulary, which is derived from words attribute values. That is, for each triple $\langle s, p, o \rangle \in E_A$ we add all words $\in o$ to W. See Fig. 5 and Ex. 2 for our running example. In its learning process, a TRM is modeled as a BN based on information from the underlying data graph.

Example 4. Fig. 4 depicts an extract of a TRM BN constructed for entity p_2 from the running example (Fig. 1). Observed variables (dark Grey) consist of entity words (e.g., $w(p_2$,"Audrey") or $w(p_2$,"Hepburn")), entity classes (e.g., `Person`(p_2)) and entity relations (e.g.,`starring`(m_1, p_2) or `spouse`(p_2, p_3)). Dependencies among observed variables are reflected by a set of hidden variables (light Grey), which are initially unobserved, but inferred during learning: the variable $\mathbf{b}(p_2)$ is a topic vector indicating the presence/absence

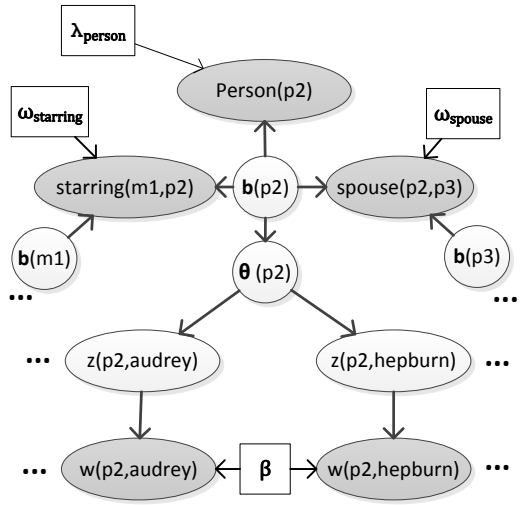


Figure 4: TRM BN extract for entity p_2: observed variables (dark Grey), hidden variables (light Grey) and TRM parameters (rectangles). Note, relation `bornIn` is not shown for space reasons.

of topics for entity p_2. Note, a relation variable also depends on a variable $\mathbf{b}$, modeling the other entity involved in the relation. This way $\mathbf{b}(p_2)$ "controls" topics selected for an entity according to structure information. In addition, $\theta(p_2)$ models the topic proportion according to $\mathbf{b}(p_2)$, whereas each a variable $z(p_2, *)$ selects a particular topic for each word by sampling over $\theta(p_2)$.

A TRM is constructed using a generative process, which is controlled via its three parameters: $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\omega}$ (shown in rectangles in Fig. 4). Hidden variables as well as parameters are inferred via a variational Bayesian learning technique as an *offline* process. Further discussion on the construction is unfortunately out of scope. For this work we apply standard TRM learning – the interested reader finds details in [23].

Important for selectivity estimation, however, are solely the learned TRM parameters that specify the topics $\mathcal{T}$ and also qualify the degree of dependency between structured data elements and topics:

(1) *Class-Topic Parameter* $\boldsymbol{\lambda}$. A TRM captures correlations between classes $\in V_C$ and hidden topics $\in \mathcal{T}$ via a global parameter $\boldsymbol{\lambda}$ that is shared among all entities in $\mathcal{G}$. $\boldsymbol{\lambda}$ is represented as a $|V_C| \times K$ matrix, where each row λ_c ($c \in V_C$) is a topic vector and each element λ_{ct} in that vector represents the weight between class c and topic t.

(2) *Relation-Topic Parameter* $\boldsymbol{\omega}$. Given K topic $\in \mathcal{T}$, the probability of observing a relation r is captured in a $K \times K$ matrix ω_r. For any two entities s/o, such that s is associated with topic t_k and o with topic t_l, the weight of observing a relation r between these entities $\langle s, r, o \rangle$ is given by an entry (k, l) in matrix ω_r (denoted by $\omega_{rt_kt_l}$). Note, a TRM provides a matrix ω for each distinct relation in $\mathcal{G}$.

Above TRM parameters are shared among all entities in the data graph. An example is given in Fig. 5.

Theorem 1. (Synopsis Space Complexity). Given K topics and a vocabulary W, a TRM requires a fixed-size space of the order of $O(|W| \cdot K + |V_C| \cdot K + |\ell_r| \cdot K^2)$.

Proof. For each topic, we store probabilities of every word in W, so the complexity of K topics is $O(|W| \cdot K)$. $\boldsymbol{\lambda}$ can be represented as a matrix $|V_C| \times K$, associating classes with topics $\in O(|V_C| \cdot K)$. Every relation is represented as a matrix $K \times K$, resulting in a total synopsis space complexity $O(|W| \cdot K + |V_C| \cdot K + |\ell_r| \cdot K^2)$ ■

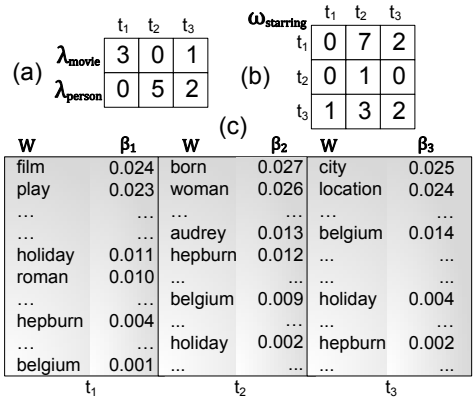


Figure 5: TRM parameters for three topics: (a) Unnormalized λ_{movie} and λ_{person} parameters over three topics. (b) ω matrix for `starring` relation (rows (columns) represent source (target) topics of the relation). (c) Selected words in three topics with their corresponding probabilities. Note, data is taken from the running example, cf. Fig. 1.

Discussion. As a data synopsis $\mathcal{S}$ for selectivity estimation a TRM offers a number of unique characteristics: (1) First, learned topics provide a low-dimensional data summary. Depending on the complexity of the structure and the amount of the textual data, a small number of topics (e.g., 50) can easily capture meaningful correlations from the data graph. Notice, while we "manually" set the number of topics for our evaluation system, our approach could be extended to learn the optimal number of topics via a non-parametric Bayesian model [9]. By means of this low-dimensional summary, a TRM provides a synopsis with linear space complexity w.r.t. the string synopsis (to be precise, linear in vocabulary W), cf. Theorem 1. (2) Each topic has a *broad coverage*, as every word in the vocabulary is covered in every topic with distinct probabilities. See β_{tw} in the running example, Fig. 5-c. Thus, in contrast to synopses based on graphical models, e.g., PRMs [12], a TRM is not restricted to small samples. Given the *contains* semantics of string predicates, sample spaces of attribute random variables can quickly "blow up". Reducing such large sample spaces via a string synopsis, as done in our previous work [20], comes at the expensive on an information loss. (3) Correlations between structure elements (classes and relations) and words $\in W$ are described in a compact way (between the classes/relations and the topics) via the structural correlation parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$. Thus, no "separate" synopses for structure and text are necessary – in contrast to [20].

Maintenance. In a real-world setting, a data synopsis must be able to handle changes in its underlying dataset. For a TRM-based synopsis dealing with changes is twofold. First, in case of *minor* variations, TRM parameters may be still allow for accurate selectivity estimation. This is due to the fact that a TRM captures dependencies between text and structured data elements via probability distributions over topics. We observed in our experiments that such probability distribution are invariant given small changes in the data. We learned TRMs from different samples (sizes) of the underlying data, however, the resulting selectivity estimations respectively topic distributions differed only marginally. Second, in case of *major* changes in the data, TRM probability parameters must be recomputed. In our experiments, TRM construction could be done in under 10 hours. However, recent work on topic models [16] has shown that this learning process can be parallelized, thereby guaranteeing a scalable TRM construction even for large data graphs. Furthermore, [2] introduced an algorithm for incremental topic model learning over text streams. Both such directions may be exploited in future work.

4.2 Selectivity Estimation Function

We address the problem of estimating $sel_{\mathcal{G}}(Q)$ via an estimation function $F_{\mathcal{S}}(Q)$, which can be decomposed as [10]:

$$F_{\mathcal{S}}(Q) = \mathcal{R}(Q) \cdot \mathcal{P}(Q)$$

Let $\mathcal{R}$ be a function $\mathcal{R} : \mathcal{Q} \to \mathbb{N}$ providing an *upper bound cardinality* for a result set for query $Q \in \mathcal{Q}$. Further, let $\mathcal{P}$ be a *probabilistic component* assigning a probability to Q that models whether or not its result is non-empty. $\mathcal{R}(Q)$ can be easily computed as product over "class cardinalities" of Q [10, 20]. More precisely, for each variable $v \in V_{Q_V}$ we bound the number of its bindings, $R(v)$, as number of entities belonging v's class: $|\{s|\langle s, type, c\rangle \in E\}|$. If v has no class, we use the number of all entities, $|V_E|$, as estimate for $R(v)$. Then, $\mathcal{R}(Q) = \prod_v R(v)$.

For the probabilistic component $\mathcal{P}$, we construct a query-specific BN (Sect. 4.2.1), and show estimation of $\mathcal{P}(Q)$ by means of this BN in Sect. 4.2.2 + 4.2.3.

4.2.1 Query-Specific BN

A query-specific BN is very similar to an "unrolled" BN given a PRM [12]. In both cases, one constructs a small BN at runtime only for the current query, by using probability distributions and dependency information from an offline data synopsis. However, our query-specific BN has unique advantages: (1) It follows a simple, yet effective fixed structure. In particular, a query-specific BN comprises one random for each query predicate, thus, no aggregations of multiple assignments to a single random variable can occur. (2) Construction does not require any marginalization, instead TRM parameters can be used directly. (3) No inferencing is needed for estimating $\mathcal{P}(Q)$ (discussed in Sect. 4.2.2 + 4.2.3). Given no inferencing and marginalization, a TRM synopsis may be kept on disk, in contrast to current PRM implementations requiring an in-memory storage.

Now, let us present the query-specific BN in more detail. We capture every query predicate as a random variable: for each class $\langle s, type, c\rangle$ and relation predicate $\langle s, r, o\rangle$, we introduce a binary random variable X_c and X_r, respectively. Similarly, for a string predicate $\langle s, a, w\rangle$, we introduce a binary random variable X_w.[4] Further, every query variable $v \in V_{Q_V}$ is considered as referring to a topic in the TRM and introduced via a *topical random variable*, X_v. However, instead of a "hard" assignment of variable X_v to a topic, X_v has a multinomial distribution over the topics. Thus, X_v captures v's "relatedness" to every topic:

Definition 2. For a set of topics $\mathcal{T}$, a query Q and its variables $v \in V_{Q_V}$, the random variable X_v is a multinomial topical random variable for v, with $\mathcal{T}$ as sample space.

Based on topical random variables, we perceive unknown query variables as topic mixtures. Then, we establish dependencies between topical random variables and random variables for class (X_c), relation (X_r) and string predicates (X_w). In order to obtain a simple network structure, we employ a fixed structure assumption:

[4]Note, attribute label a is omitted in the notation, since topic models do not distinguish attributes.

Definition 3 (Topical Independence Assump.)**.** Given a query Q and its variables V_{Q_V}, the probability of every query predicate random variable, X_i, $i \in \{w, c, r\}$, is independent from any other predicate random variable, given its parent topical random variable(s), $\mathbf{Pa}(X_i) \subseteq \{X_v\}_{v \in V_{Q_V}}$.

The topical independence assumption lies at the core of our `TopGuess` approach. (1) It considers that query predicate probabilities depend on (and governed by) the topics of their corresponding query variables. In other words, during selectivity estimation we are looking for a specific ("virtual") binding to each query variable, whose topic distribution is represented in its corresponding topical random variable (initially unknown) and determined by query predicates. (2) It allows to model the probability $\mathcal{P}(Q)$ in a simple query-specific BN (cf. Fig. 6). Here, the probability of observing a query predicate is solely dependent on the topics of query variables, which enables us to handle dependencies among query predicates in a simplistic manner (Sect. 4.2.2). Note, a generic query-specific BN is given in Fig. 6-a, while Fig. 6-b depicts a query-specific BN for our running example. Our network structure allows a *valid* BN, as the following theorem holds:

Theorem 2. The query-specific BN constructed according to the topical independence assumption is acyclic.

Proof Sketch. BN parts resembling class and string variables form a forest of trees – each tree has depth one. Such trees are combined via relation predicate variables, which have no children (cf. Fig. 6-a). Thus, no cycle can be introduced ■
Intuitively, the query probability $\mathcal{P}(Q)$ can be written as:

$$\mathcal{P}(Q) = \mathcal{P}\left(\mathbf{X}_w = \mathbf{T} \wedge \mathbf{X}_c = \mathbf{T} \wedge \mathbf{X}_r = \mathbf{T}\right) \qquad (1)$$

Let $\mathbf{X}_w$, $\mathbf{X}_c$ and $\mathbf{X}_r$ refer to sets of string, class and relation predicate variables in a query-specific BN for Q. For above equation, we need the topic distributions of query variables. Given such distributions (explained in Sect. 4.2.3), we can estimate predicates probabilities and Eq. 1 with a time complexity, which is independent of synopsis $\mathcal{S}$: $O(|Q| \cdot K)$, where $|Q|$ and K is the number of query predicates and topics, respectively. As topical random variables X_v are hidden, we learn such distributions from observed predicate variables. In Sect. 4.2.2, we first discuss parameter learning for observed random variables, given topical random variables, i.e., query variable topics, and subsequently present learning of hidden topical random variables (Sect. 4.2.3).

4.2.2 Query Predicate Probabilities

Query predicates probabilities are influenced by their corresponding topical random variables (in a query-specific BN) as well as their prior probabilities in the underlying TRM synopsis. Thus, we can formulate the conditional probability of each query predicate variable (X_c, X_r and X_w) by incorporating topic distributions of query variables (i.e., topical random variables) with prior probabilities estimated using TRM parameters (i.e., $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$). That is, probabilities for query predicate variables are obtained as follows:

Class Predicate Variables. Adhering to the topical independence assumption, the probability of observing a class, $P(X_c = \mathbf{T})$, is only dependent on its topical variable X_s. Here, we use TRM parameter $\boldsymbol{\lambda}$ to obtain the weight λ_{ct}, indicating the correlation between topic t and class c. The probability of observing class c is given by:

$$P(X_c = \mathbf{T}|X_s, \boldsymbol{\lambda}) = \sum_{t \in \mathcal{T}} P(X_s = t) \frac{\lambda_{ct}}{\sum_{t' \in \mathcal{T}} \lambda_{ct'}}$$

Example 5. Fig. 6-b illustrates two class variables, X_{movie} and X_{person}, which are dependent on the random variables X_m and X_p, respectively. For computing query probabilities, $P(X_{movie} = \mathbf{T})$ and $P(X_{person} = \mathbf{T})$, the corresponding TRM parameters λ_{movie} and λ_{person} (Fig. 5) are used. For instance, given $P(X_m = t_1) = 0.6$, $P(X_m = t_2) = 0.1$, and $P(X_m = t_3) = 0.3$, we have: $P(X_{movie} = \mathbf{T}) = 0.6 \cdot {}^3\!/_4 + 0.1 \cdot {}^0\!/_4 + 0.3 \cdot {}^1\!/_4 = 0.525$.

Relation Predicate Variables. Every relation predicate $\langle s, r, o \rangle$ connects two query variables, for which there are corresponding topical variables X_s and X_o. The variable X_r (representing this relation predicate) solely dependents on the topics of X_s and X_o. The dependency "strength" between r and topics of these two variables is captured by parameter ω_r in the TRM. Using ω_r, the probability of observing relation r is:

$$P(X_r = \mathbf{T}|X_s, X_o, \omega_r) = \sum_{t,t' \in \mathcal{T}} \frac{P(X_s = t)\ \omega_{rtt'}\ P(X_o = t')}{\sum_{t''t''' \in \mathcal{T}} \omega_{rt''t'''}}$$

Example 6. In Fig. 6-b, there are two relation predicate variables: $X_{starring}$ and X_{bornin}. Each of them is dependent on two topical variables, e.g., X_m and X_p condition $X_{starring}$. Probability $P(X_{starring} = \mathbf{T})$ is estimated via matrix $\omega_{starring}$ (Fig. 5).

String Predicate variables. For each string predicates $\langle s, a, w \rangle \in Q$, there is a random variable X_w. The parameter β_{tw} provided by a TRM represents the probability of observing a word w given a topic t. Thus, the probability $P(X_w = \mathbf{T})$ is calculated as probability of observing w, given the topics of its query variable X_s and $\beta_{1:K}$:

$$P(X_w = \mathbf{T}|X_s, \beta_{1:K}) = \sum_{t \in \mathcal{T}} P(X_s = t) \frac{\beta_{tw}}{\sum_{t' \in \mathcal{T}} \beta_{t'w}}$$

Example 7. Fig. 6-b depicts four string predicate variables, needed for the three string predicates comprised in our query (Fig. 2). Note, for `name` = "Audrey Hepburn" two variables X_{audrey} and $X_{hepburn}$ are necessary. Given $P(X_m)$ as in the example above, $P(X_{holiday} = \mathbf{T})$ is:

$$P(X_{holiday} = \mathbf{T}) = 0.6 \cdot \frac{0.011}{0.017} + 0.1 \cdot \frac{0.002}{0.017} + 0.3 \cdot \frac{0.004}{0.017} = 0.47$$

4.2.3 Learning Topics of Query Variables

The core idea of `TopGuess` is to find an optimal distribution of topical random variables, so that the joint probability of the query-specific BN (Eq. 1) is maximized. Thus, as a final step, we learn parameters for our initially unobserved topical random variables, based on observed predicate variables. For computing these parameters, we first introduce a set of topic parameters θ_{vt} for each topical random variable X_v. $\boldsymbol{\theta} = \{\theta_{vt} | v \in V_{Q_V}, t \in \mathcal{T}\}$ denotes the set of parameters for all topical variables. As before, $\mathbf{X}_w$, $\mathbf{X}_c$ and $\mathbf{X}_r$ denote string, class and relation predicate variables in a query-specific BN. Then, we find parameters $\boldsymbol{\theta}$ for topic variables, which maximize the log-likelihood of Eq. 1. The optimization problem is:

$$\arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)$$

where $\ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)$ is the log-likelihood defined as:

$$\begin{aligned} \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r) &= P(\mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r|\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\omega}, \boldsymbol{\lambda}) \\ &= \sum_v \sum_{X_w \in \mathbf{X}_w^v} \log P(X_w|X_v, \boldsymbol{\beta}) \end{aligned}$$

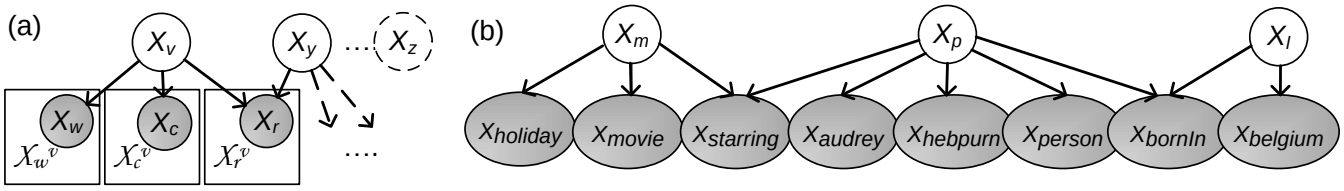


Figure 6: (a) Generic query-specific BN in plate notation. Notice, string predicate variables (X_w), class predicates variables (X_c), and relation predicate variables (X_r) are only dependent on their topical random variable X_v/X_y. (b) A query-specific BN for query in Fig. 2 with 3 topical variables (e.g., X_m), 2 class predicate variables (e.g., X_{movie}), 2 relation predicate variables (e.g., $X_{starring}$) and 4 string predicate variables (e.g., $X_{holiday}$). Observed variables (dark Grey) are independent from each other and only dependent on hidden topical random variables (light Grey) – adhering to the topical independence assumption (Def. 3).

$$+ \sum_v \sum_{X_c \in \mathbf{X}_c^v} \log P(X_c|X_v, \boldsymbol{\lambda})$$
$$+ \sum_{v,y} \sum_{X_r \in \mathbf{X}_r^{v,y}} \log P(X_r|X_v, X_y, \boldsymbol{\omega})$$

where $\mathbf{X}_w^v$ and $\mathbf{X}_w^v$ is the set of all string and class random variables with a parent X_v. $\mathbf{X}_r^{v,y}$ is the set of all relation random variables with parents X_v and X_y. We use gradient ascent optimization to learn the parameters. First, we parametrize each $P(X_v = t)$ with θ_{vt} as $P(X_v = t) = \frac{e^{\theta_{vt}}}{\sum_{t' \in \mathcal{T}} e^{\theta_{vt'}}}$ to obtain a proper probability distribution over the topics. Obtaining the gradient requires dealing with the log of the sum over the topics of each topical variable. Therefore, we make use of theorem [12]:

Theorem 3. Given a BN and $\mathcal{D} = \{\mathbf{o}[1], \ldots, \mathbf{o}[M]\}$ as a partially observed dataset. Further, let X be a random variable in that BN, and let $\mathbf{Pa}(X)$ denote its parents. Then:

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathcal{D})}{\partial P(x|pa)} = \frac{1}{P(x|pa)} \sum_{m=1}^{M} P(x, pa|\mathbf{o}[m], \boldsymbol{\theta}),$$

which provides the form of the gradient needed. Now, the gradient of the log-likelihood w.r.t. the parameter θ_{vt} can be stated as

$$\frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial \theta_{vt}} = \frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial P(X_v = t)} \frac{\partial P(X_v = t)}{\partial \theta_{vt}}.$$

The first part of the gradient can be obtained using Theorem 3:

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial P(X_v = t)} &= \frac{1}{P(X_v = t)} \Bigg(\sum_{X_w \in \mathbf{X}_w^v} P(X_v = t|X_w, \boldsymbol{\beta}) \\ &+ \sum_{X_c \in \mathbf{X}_w^v} P(X_v = t|X_c, \boldsymbol{\lambda}) \\ &+ \sum_y \sum_{X_r \in \mathbf{X}_r^{v,y}} P(X_v = t|X_r, X_y, \boldsymbol{\omega}) \Bigg)\end{aligned}$$

Using the Bayes rule we have:

$$\begin{aligned}&\frac{\partial \ell(\boldsymbol{\theta} : \mathbf{X}_w, \mathbf{X}_c, \mathbf{X}_r)}{\partial P(X_v = t)} = \\ &\sum_{X_w \in \mathbf{X}_w^v} \frac{P(X_v = t) P(X_w|\boldsymbol{\beta}, t)}{\sum_{t'} P(X_v = t') P(X_w|\boldsymbol{\beta}, t')} + \\ &\sum_{X_c \in \mathbf{X}_w^v} \frac{P(X_v = t) P(X_c|\boldsymbol{\lambda}, t)}{\sum_{t'} P(X_v = t') P(X_c|\boldsymbol{\lambda}, t')} + \\ &\sum_y \sum_{X_r \in \mathbf{X}_r^{v,y}} \frac{P(X_v = t) \sum_{t'} P(X_r|X_y, \boldsymbol{\omega}, t')}{\sum_{t''} P(X_v = t'') \sum_{t'''} P(X_r|X_y, \boldsymbol{\omega}, t''')}\end{aligned}$$

Finally, the second part of the gradient is given by:

$$\frac{\partial P(X_v = t)}{\partial \theta_{tv}} = \frac{e^{\theta_{tv}} \sum_{t' - t} e^{\theta_{t'v}}}{(\sum_{t'} e^{\theta_{t'v}})^2}$$

5. EVALUATION

We present experiment results to analyze the effectiveness (Sect. 5.1, **Req.1+3**) and the efficiency (Sect. 5.2, **Req.2+4+5**) of selectivity estimation using `TopGuess`. By means of the former, we wish to compare the quality of estimates. Previous work has shown that estimation quality is of great importance for many use cases, most notably for query optimization [19]. The latter aspect targets the applicability of our solution towards real-world systems.

Building up on our recent work [20], we compare `TopGuess` with two different kinds of baselines: (1) approaches that assume independence among string predicates as well as between them and structured query predicates (`ind`). (2) Our previous system [20], i.e., a PRM-based approach with integrated string synopses for dealing with string predicates (`bn`). Our results are promising: as average over both datasets a gain of 20% w.r.t. the best baseline could be achieved by `TopGuess`. In terms of efficiency, `TopGuess` resulted in similar performances as the baselines. However, we noted that its efficiency was much less influenced by query respectively synopsis size than `ind` or `bn` approaches.

Systems. As baselines, we consider two categories of approaches. (1) For selectivity estimation, string predicates are combined with structured predicates via an *independence assumption* (`ind`). More precisely, the selectivity of string and structured predicates is estimated using a string synopsis and histograms, respectively. Obtained probabilities were combined in a greedy fashion assuming independence. (2) Multiple query predicates are combined relying on a BN data synopsis (`bn`). That is, we reuse our work on PRM approaches for selectivity estimation on text-rich data graphs [20]. Note, Ex. 3 depicts an examplary PRM. The `bn` baseline handles the problem of multiple value assignments to a single random variable via aggregation functions. We use a *stochastic mode* aggregation, which essentially uses all assignments, but weights each one with its frequency [18]. As string synopsis function ν we exploited work on n-gram string synopses [21]. A synopsis based on n-grams reduces the common representation space by using a predefined decision criterion to dictate which n-grams to in-/exclude. Note, we refer to discarded n-grams as "missing". That is, a synopsis represents a subset of all possible n-grams occurring in the data graph. A simplistic strategy is to choose random n-gram *samples* from the data. Another approach is to construct a *top-k* n-gram synopsis. For this, n-grams are extracted from the data graph together with their counts.

Then, the k most frequent n-grams are included in the representation space. Further, a *stratified bloom filter* synopsis has been proposed [21], which uses bloom filters as a heuristic map that projects n-grams to their counts. We use these three types of n-grams synopses in our experiments. The probability for "missing" n-grams cannot be estimated with a probabilistic framework, as such strings are not included in a sample space. To deal with this case, a string predicate featuring a missing n-gram is assumed to be independent from the remainder of the query. Then, its probability can be estimated based on various heuristics. We employ the *leftbackoff* strategy, which finds the longest known n-gram that is the pre- or postfix of the missing keyword and estimates its probability based on the statistics for that pre- and postfix [21]. Combining string synopses with our two categories of baselines results in six different systems: $\texttt{ind}_{\text{sample}}$, $\texttt{ind}_{\text{top-}k}$ and $\texttt{ind}_{\text{sbf}}$ rely on the independence assumption, $\texttt{bn}_{\text{sample}}$, $\texttt{bn}_{\text{top-}k}$ and $\texttt{bn}_{\text{sbf}}$ represent BN approaches.

Data. We employ two real-world RDF datasets for evaluation: DBLP[5] comprising computer science bibliographies and IMDB [6] featuring movie information. For both datasets, we could extract large vocabularies containing $25,540,172$ and $7,841,347$ 1-grams from DBLP and IMDB, respectively. See also Table 1 for an overview. Notice, while IMDB as well as DBLP both feature text-rich attributes like `name`, `label` or `info`, they differ in their overall amount of text: IMDB comprises large text values, e.g., associated via `info`, DBLP, however, holds much less unstructured data. On average an attribute in DBLP contains only 2.6 1-grams with a variance of 2.1, in contrast to IMDB with 5.1 1-grams given a variance 95.6. Further, also the attribute with the most text associated is larger, having 28.3 1-grams, for IMDB, than for DBLP with 8.1 1-grams (cf. Table 1). Our hypothesis is that these differences will be reflected in different degrees of correlations between text and structured data. Moreover, we are interested in comparing performance of `bn` and `TopGuess` w.r.t. varying amounts of texts. In particular, given `TopGuess` using a topic model data synopsis, we wish to analyze its effectiveness in such settings.

Queries. For our query load we reuse existing work on keyword search evaluation [14, 6]. We form queries adhering to our model by constructing graph patterns, comprising string, class, and relation predicates that correspond to the given query keywords and their structured results. We generated 54 DBLP queries based on "seed" queries reported in [14]. That is, for each query in [14], we replaced its keyword constants with variables, evaluated such seed queries, and generated new queries by replacing a keyword variable with one of its randomly selected bindings. Additionally, 46 queries were constructed for IMDB based on queries taken from [6]. We omitted 4 queries in [6], as they could not be translated to conjunctive queries. Overall, our load features queries with $[2, 11]$ predicates in total: $[0, 4]$ relation, $[1, 7]$ string, and $[1, 4]$ class predicates. As our query model allows solely single keywords to be used, we treat string predicates with phrases as several predicates. During our analysis, we use the number of predicates as an indicator for query complexity. We expect queries with a larger number of predicates to be more "difficult" in terms effectiveness and efficiency. Further, we expect correlations between query predicates to have a strong influence on system effectiveness. Note, we observe during structure learning of the `bn` baseline systems different degrees of correlations in DBLP and IMDB. More precisely, we noticed that there are more correlated predicates in IMDB, e.g., `name` (class `Actor`) and `title` (class `Movie`), than in DBLP. Query statistics and a complete query listing is given in our report [22].

[5] `http://knoesis.wright.edu/library/ontologies/swetodblp/`

Table 1: Dataset Statistics

	IMDB	DBLP
# Triples	7,310,190	11,014,618
# Resources	1,673,097	2,395,467
# Total 1-grams	7,841,347	25,540,172
# Avg. 1-grams Mean	5.1	2.6
# Avg. 1-grams Variance	95.6	2.1
Max. attr. # avg. 1-grams	28.3	8.1
# Attributes	10	20
# Relations	8	18
# Classes	6	18

Table 2: Storage Space (MByte)

	IMDB		DBLP	
	Data			
Disk	1600		5800	
	Data Synopsis			
	`bn` & `ind`	`TopGuess`	`bn` & `ind`	`TopGuess`
Mem.	$\{2, 4, 20, 40\}$	≤ 0.1	$\{2, 4, 20, 40\}$	≤ 0.1
Disk	0	281.7	0	229.9

Synopsis Size. Using the same configurations as in [20], we employ different synopsis sizes for our baselines `ind` and `bn`. The factor driving the overall synopses size for `ind` and `bn` is their string synopsis size, i.e., the size of their common representation space $|\mathcal{C}|$. This effect is due to $|\mathcal{C}|$ determining the size of the (conditional) probability distribution in $\texttt{ind}_*$ ($\texttt{bn}_*$). CPDs, however, are very costly in terms of space, while other statistics, e.g., the BN structure, are negligible. We varied the number of 1-grams captured by the top-k and sample synopsis: #1-grams per attribute $\in \{0.5K, 1K, 5K, 10K\}$. For the sbf string synopsis, we captured up to $\{2.5K, 5K, 25K, 50K\}$ of the most frequent 1-grams for each attribute and varied the bloom filter sizes, resulting in similar memory requirements. Note, the sbf systems featured all 1-grams occurring in our query load. Except for `TopGuess`, all systems load the synopsis into main memory. To be more precise, only $\texttt{bn}_*$ approaches require the synopsis to be in-memory for inferencing. For comparison, however, we also load statistics for $\texttt{ind}_*$ approaches. As shown in Table 2, different string synopses (sizes) translate to approaches consuming $\{2, 4, 20, 40\}$ MByte of memory. As opposed to $\texttt{bn}_*$ and $\texttt{ind}_*$ approaches, `TopGuess` keeps a large topic model at disk, and constructs a small, query-specific BN in memory at runtime (memory consumption $\leq$ 100 KBytes on average). Thus, all query-independent statistics remain on the hard-disk. The large disk size ($\in [220 - 280]$ MByte) of `TopGuess` comes from the use of all 1-grams available in the data. Such a fine-grained model enables very accurate estimations, as we will show in the following. However, it still preforms comparable in terms of efficiency. Table 2 shows an overview of the main memory and disk usage required by the different systems.

Implementation and Offline Learning. For baselines $\texttt{bn}_*$ and $\texttt{ind}_*$ [20], we started by constructing string synopses. Each synopsis, including sbf-based synopses, was learned in $\leq 1h$. As $\texttt{bn}_*$ and $\texttt{ind}_*$ use the same probability distributions (BN parameters), parameters were trained together. For $\texttt{bn}_*$ we use a PRM construction as done in [20]. That is, we capture un-/structured data elements using random variables and learn correlations between them, thereby forming a network structure. For efficient selectivity estimation the network is reduced to a "lightweight" model,

capturing solely the most important correlations. Then, we calculate the model parameters (CPDs) based on frequency counts. For $\texttt{ind}_*$ systems, we do not need the model structure and merely keep the marginalized parameters. Structure and parameter learning combined took in the worst case up to three hours. The structure and the parameters are stored in a key-value store outside the database system – both were loaded at start-up. Depending on the synopsis size, loading the model into memory took up to $3s$. The inferencing needed by the $\texttt{bn}_*$ systems for selectivity estimation is done using a Junction tree algorithm [20]. `TopGuess` exploits a TRM for which we use a standard learning procedure from [23]. For TRM learning, a reasonable sample of the dataset is sufficient, instead of learning over the whole dataset. We sampled up to 30K entities per class from each dataset, to ensure that all classes and relations are equally represented. The number of topics in a TRM is an important factor, determining which correlations are discovered from the underlying dataset. That is, a "sufficient" number of topics is needed to correlate the textual information to a heterogeneous set of classes and predicates. Note, a TRM is a supervised topic model which handles the sparseness of these topics correlated to structure information. In other words, some topics can be correlated with many structure elements, whereas others are not. We experimented with a varying number of topics $\in [10 - 100]$ and found that, for datasets employed in our evaluation, 50 topics are enough to capture all important correlations. Notice, a TRM may easily be extended to determine the number of topics based on the data in an unsupervised fashion, by using a non-parametric Bayesian model [9]. The TRM learning took up to 10 hours, and the resulting models were stored in an inverted index on hard disk (cf. Table 2). At query time, we employed a greedy gradient ascent algorithm for estimation of the probability distributions of hidden topic variables. To avoid local maxima, we used up to 10 random restarts.

We implemented all systems and algorithms using Java 6. Experiments were run on a Linux server with two Intel Xeon 5140 CPUs (each with 2 cores at 2.33GHz), 48 GB RAM (with 16 GB assigned to the JVM), and a RAID10 with IBM SAS 148GB 10k rpm disks. Before each query execution, all OS caches were cleared. The presented values are averages collected over five runs.

5.1 Selectivity Estimation Effectiveness

We employ the *multiplicative error* metric (me) [8] for measuring estimation effectiveness.

$$me(Q) = \frac{\max(sel(Q), \overline{sel}(Q))}{\min(sel(Q), \overline{sel}(Q))}$$

with $sel(Q)$ and $\overline{sel}(Q)$ as exact and approximated selectivity for Q, respectively. Intuitively, me gives the factor to which $\overline{sel}(Q)$ under- or overestimates $sel(Q)$.

Overall Results. Fig. 7-a, -b and -e, -f show the multiplicative error vs. synopsis size for DBLP and IMDB, respectively. As expected, baseline system effectiveness strongly depends on the (string) synopsis size. That is, for small synopses ≤ 20 MByte, $\texttt{ind}_*$ and $\texttt{bn}_*$ performed poorly. We explain this with the information loss, due to omitted 1-grams in the common representation space (**Req.3**). That is, the employed string synopsis traded space for accuracy, and heuristics had to used for probability estimation. In fact, in case of $\texttt{bn}_*$, the information loss is aggravated as missed keyword can not be added to an unrolled BN, instead one must assume independence between such a string predicate and the remainder of the query [20]. `TopGuess`, on the other hand, did not suffer from this issue, as all query-independent statistics could be stored at disk, and solely the query-specific BN was loaded at runtime. Thus, `TopGuess` could exploit very fine-grained probabilities, and omitted any kind of heuristics. We observed that, on average, `TopGuess` could reduce the error of the best performing $\texttt{bn}_{\text{sbf}}$ by 20%, and the best system using the independence assumption, $\texttt{ind}_{\text{sbf}}$, by 93%. As in previous work [20], different string synopses in $\texttt{ind}_*$ and $\texttt{bn}_*$, yielded different estimation effectiveness results. Sampling-based systems were outperformed by systems using top-k n-grams synopses, which in turn, performed worse than sbf-based approaches. These drastic misestimates come from query keywords being "missed" in the string synopsis. Thus, we can see estimation quality being strongly influenced by fine-grained, accurate string probabilities. We argue that such results clearly show the need for a data synopsis that is conceptually different from current approaches – allowing for effective on-disk storage of large statistics (**Req.2**).

Synopsis Size. Fig. 7-a/-e shows estimation errors w.r.t. synopsis size and memory consumption, respectively. One can see that, compared to other approaches, `TopGuess` memory usage is a small constant ≤ 0.1 MByte. This extremely compact BN can be explained with `TopGuess` only using random variables that either are *binary* or have a sample space bound by the number of topics, K. Note, both such factors are constant in the overall data synopsis size. Further, we noted the performances of the sbf-based approaches to be fairly stable, while top-k and especially sample-based n-gram systems were strongly affected by the (string) synopsis size. That is, both performed poorly given small synopsis size ($\leq$ 4 MByte), with increasing synopses, however, the performances of the top-k and sbf-based approaches converge. Overall, while sbf-based approaches proofed to be an effective strategy, they also suffer from limit (in-memory) space, and thus must discard words in the vocabulary. Instead, `TopGuess` resolves the issue of missing n-grams completely: the query-independent TRM stored on disk captures statistics for all n-grams. At runtime, `TopGuess` retrieves the necessary n-grams for a particular query, and constructs its query-specific BN (**Req.3**).

Correlations and Degree of Text Data. Following our initial hypothesis, we found performances for IMDB and DBLP to vary. That is, given IMDB `TopGuess` and $\texttt{bn}_{\text{sbf}}$ could reduce errors of the $\texttt{ind}_{\text{sbf}}$ approach by 99% and 93%, respectively. On the other hand, for DBLP improvements were much smaller. These differences are due to the varying degree of correlations in our two datasets. While learning the BNs for `bn`, we observed significantly less correlations in DBLP than in IMDB. More importantly, many of the DBLP queries include string predicates such as `name` and `label`, which we could not observe any correlations. For such queries, the probabilities obtained by $\texttt{bn}_*$ were close to the ones computed by $\texttt{ind}_*$. At the same time, we noted that the degree of correlation between un-/structured data, is greatly influenced by the number/size of text values. Essentially, we noted that given attributes with more text values, more correlations among them respectively structured data elements tend to occur. For instance, given queries involving attribute `name` in DBLP, with only 2.4 1-grams (variance: 2.1 1-grams, cf. Table 1), with measured over 30 % less correlations than for queries on IMDB with attribute `info`. These effects confirm our hypothesis that the degree of text influences correlations, which in turn drive system effectiveness. However, compared to $\texttt{bn}_*$, `TopGuess` relies

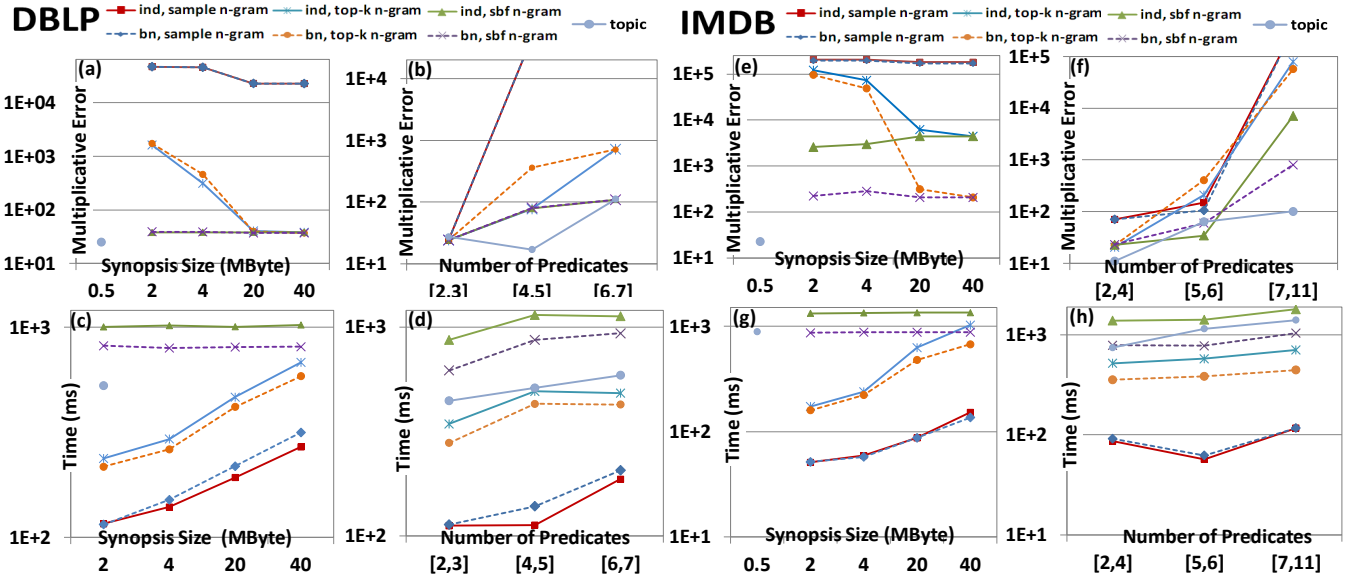


Figure 7: Evaluation Results for DBLP and IMDB.

on a more fine-grained BN: while $\texttt{bn}_*$ exploits correlations observed in the data graph before query time, `TopGuess` utilizes all query predicate correlations via a query-specific BN at runtime. Thus, a `TopGuess` BN is able to capture even "minor" correlations, which may have been discarded by $\texttt{bn}_*$ in favor of a compact structure. Note that previous works on PRMs for selectivity estimation [19, 20], aim at a "lightweight" model structure, i.e., dependency informations is traded for efficient storage and inferencing. Such trade-offs, however, are not necessary for `TopGuess`. Thus, even for the less correlated dataset DBLP, `TopGuess` outperforms the baseline $\texttt{ind}_{sbf}$ and $\texttt{bn}_{sbf}$ by 20% and 15%. We argue that this result also confirms the general applicability of `TopGuess`. Even for "little" textual data, the TRM synopsis was able to capture meaningful topics, leading to accurate probability estimates for query-specific BNs (**Req.1**).

Query Size. In Fig. 7-b/-f we depict multiplicative error (average over synopsis sizes) vs. number of query predicates. As expected, estimation errors increase for all systems in the number of query predicates. For our baselines, we explain this behavior via: (1) given a higher number of predicates chances of "missing" a keyword increase, and (2) when missing an n-grams, the error is propagated to the estimate for the remainder of the query (which might have been fine otherwise). However, while the `TopGuess` approach also led to more misestimates for larger queries, the degree of this increase was smaller. In particular, considering highly correlated queries for IMDB with size $\in [7-11]$, we can observe (Fig. 7 -f) `TopGuess` to perform much more stable than $\texttt{bn}_*$ or $\texttt{ind}_*$. As observed in [20], we also noticed misestimates of `bn` due to inaccurate stochastic value aggregation. This effect led to $\texttt{ind}_*$ outperforming $\texttt{bn}_*$ for some queries. `TopGuess` does not suffer from such a problem, because its random variables are "predicate-specific", i.e., we construct one single random variable for each query predicate at runtime. Overall, compared to $\texttt{bn}_*$ respectively $\texttt{ind}_*$, `TopGuess` yielded the most accurate and stable performance.

5.2 Selectivity Estimation Efficiency

Let us analyze the estimation efficiency for varying synopses sizes (Fig. 7-c/-g) and query complexities (Fig. 7-d/-h). As `TopGuess` uses a query-specific model, its times comprise model loading and construction as well as learning. For `bn` and `ind`, the reported times represent solely the inference task, i.e., time for model construction and loading have been omitted.

Overall Results. As noted in [20], we also observed that for `bn`/`ind` not BN inferencing, but the string synopsis was driving the performance. Intuitively, the more n-gram were missed, the "simpler" and the more efficient these systems became. However, such performance gains come at the expense of estimation effectiveness – the fastest baseline system relied on sample-based synopses. In fact, the very same systems performed worst in terms of effectiveness.

Comparing the two systems with best effectiveness, i.e., `TopGuess` and $\texttt{bn}_{\text{sbf}}$, `TopGuess` led to a better performance by 29Ḣowever, in comparison to top-k systems, `TopGuess` resulted in a performance decrease of 28%. We explain these performance drawbacks with the time-consuming disk I/O, which was needed for loading the necessary statistics.

However, `TopGuess` performance results are still promising: (1) Its efficiency it is not driven by the overall synopsis size. That is, while `bn` and `ind` clearly outperform `TopGuess`, given small synopses $\leq$ 4 MByte, `TopGuess` results are better respectively comparable for synopses $\geq$ 20 MByte. We expect such effects to be even more drastic for "very large" `bn` (`ind`) synopses $\gg$ 100 MByte. (2) As we will discuss below, we also found that `TopGuess` performance was much less driven by query size. Considering both aspects, `TopGuess` guarantees a much more "stable" behavior (**Req.5**).

Synopsis Size. Fig. 7-c/-g shows selectivity estimation time vs. synopsis size. For baseline systems we can see a strong dependency between synopsis size and their runtime behavior (**Req.5** fails). While `bn` and `ind` reach high efficiency for synopses $\leq$ 4 MByte, there performance decreases rapidly with synopses $\geq$ 20 MByte. Note, sbf-based approaches, are an exception, as there computational costs are determined by bloom filters and not their overall number of 1-grams. `TopGuess`, does not suffer from this issue. As our approach does not require any marginalization or inferencing, constructing a query-specific BN and computing its joint probability is *independent from the size of the TRM*.

Query Size. We observe that for all systems estimation times increase with query size (cf. Fig. 7-d/-h). However, as `TopGuess` exploits an extremely compact query-specific BN, we expect it's performance to be much less influenced

by query size. To confirm this we compared the *standard deviation* of the estimation time between `TopGuess` and $\texttt{bn}_*$ w.r.t. different sizes. The standard deviations was $82,48$ ms and $213,48$ ms for `TopGuess` and $\texttt{bn}_*$, respectively. The low deviation for `TopGuess` indicates that the required I/O and learning times varied little w.r.t. query/synopsis size. For $\texttt{bn}_*$, however, its high variance suggests that the performance is strongly affected.

6. RELATED WORK

For selectivity estimation on structured data, works exploit *table-level* data synopses, which capture attributes within the same table (e.g., [15]). Other approaches focus on *schema-level* synopses, which are not restricted to a single table, but captures attributes and relations: graph synopses [17], join samples [1], and graphical models [10, 19, 20].

In contrast to `TopGuess`, such approaches do not summarize correlations in unstructured data. In fact, in our recent work [20] we only loosely integrated graphical models and string synopses. However, [20] does not provide a uniform framework. Further, it suffers from the same problems as previous PRM-based solution [10, 19]. We discussed their drawbacks in depth in our framework, Sect. 3, as well as evaluation, Sect. 5, part.

For estimating selectivities of string predicates, on the other hand, language models and other machine learning/NPL techniques have been utilized [5, 11, 13, 21]. Here, some works aim at substring respectively fuzzy string matching [5, 13], while other approaches target "extraction" operators, e.g., dictionary-based operators [21].

However, these approaches do not consider dependencies among various string predicates and/or with query predicates for structured data. In contrast, we present a *holistic approach* for hybrid queries.

7. CONCLUSION

We aim at a holistic system for selectivity estimation of hybrid queries. For this, we presented a novel approach, *TopGuess*, which enables effective and efficient estimations over RDF data. We conducted experiments on real-world datasets, which featured two kinds of baselines. (1) Approaches using independence assumptions, `ind`, and (2) solutions building up on PRM approaches, `bn`. Our results suggest that correlations, if present in the data, drive the selectivity estimation effectiveness. In particular, in the skewed IMDB dataset we observed `bn` baselines to outperform the `ind` approaches, by 12% on average. *TopGuess* led to even more accurate estimates and allowed an additional gain of 20% over the best `bn` system. Considering efficiency aspects, *TopGuess* performed comparable to the baselines.

8. REFERENCES

[1] S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *SIGMOD*, pages 275–286, 1999.

[2] A. Banerjee and S. Basu. Topic models over text streams: A study of batch and online unsupervised learning. In *SDM*, 2007.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] J. Chang and D. Blei. Relational topic models for document networks. In *AIStats*, 2009.

[5] S. Chaudhuri, V. Ganti, and L. Gravano. Selectivity estimation for string predicates: Overcoming the underestimation problem. In *ICDE*, pages 227–238, 2004.

[6] J. Coffman and A. C. Weaver. A framework for evaluating database keyword search strategies. In *CIKM*, pages 729–738, 2010.

[7] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405, 1990.

[8] A. Deshpande, M. N. Garofalakis, and R. Rastogi. Independence is good: Dependency-based histogram synopses for high-dimensional data. pages 199–210, 2001.

[9] F. Doshi, K. Miller, J. V. Gael, and Y. W. Teh. Variational inference for the indian buffet process. *Journal of Machine Learning Research*, 5:137–144, 2009.

[10] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *SIGMOD*, pages 461–472, 2001.

[11] L. Jin and C. Li. Selectivity estimation for fuzzy string predicates in large data sets. In *VLDB*, pages 397–408, 2005.

[12] D. Koller and N. Friedman. *Probabilistic graphical models*. MIT press, 2009.

[13] H. Lee, R. T. Ng, and K. Shim. Extending q-grams to estimate selectivity of string matching with low edit distance. In *VLDB*, pages 195–206, 2007.

[14] Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, and K. Li. Spark2: Top-k keyword query in relational databases. *IEEE Trans. Knowl. Data Eng.*, 23(12):1763–1780, 2011.

[15] V. Poosala, P. Haas, Y. Ioannidis, and E. Shekita. Improved histograms for selectivity estimation of range predicates. *SIGMOD*, pages 294–305, 1996.

[16] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proc. VLDB Endow.*, 3(1-2):703–710, Sept. 2010.

[17] J. Spiegel and N. Polyzotis. Graph-based synopses for relational selectivity estimation. In *SIGMOD*, pages 205–216, 2006.

[18] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *IJCAI*, pages 870–878, 2001.

[19] K. Tzoumas, A. Deshpande, and C. S. Jensen. Lightweight graphical models for selectivity estimation without independence assumptions. *PVLDB*, 4:852–863, 2011.

[20] A. Wagner, V. Bicer, and T. D. Tran. Selectivity estimation for hybrid queries over text-rich data graphs. In *EDBT*. ACM, 2013.

[21] D. Z. Wang, L. Wei, Y. Li, F. Reiss, and S. Vaithyanathan. Selectivity estimation for extraction operators over text data. In *ICDE*, pages 685–696, 2011.

[22] xxx. Topic-based selectivity estimation for text-rich data graphs - report. Technical report. `http://db.tt/6Wur5Fbs`.

[23] xxx. Trm - learning dependencies between text and structure with topical relational models. Technical report. `http://db.tt/IKlpH2WV`.

[24] J. Zeng, W. K. Cheung, C.-h. Li, and J. Liu. Multirelational topic models. In *ICDM*, 2009.